SOR2: QUESTIONÁRIO

Isaac Mota Bandeira

1. Como definir um volume no Docker Compose para persistir os dados do banco de dados PostgreSQL entre as execuções dos containers?
   Para definir um volume no Docker Compose e persistir os dados do banco de dados PostgreSQL entre as execuções dos containers, você pode usar a chave *volumes* no arquivo *docker-compose.yml.*

2. Como configurar variáveis de ambiente para especificar a senha do banco de dados PostgreSQL e a porta do servidor Nginx no Docker Compose?
   Para configurar variáveis de ambiente no Docker Compose e especificar a senha do banco de dados PostgreSQL e a porta do servidor Nginx, você pode usar a chave *environment* dentro do serviço correspondente no arquivo *docker-compose.yml.*

3. Como criar uma rede personalizada no Docker Compose para que os containers possam se comunicar entre si?
   Para criar uma rede personalizada no Docker Compose e permitir que os contêineres se comuniquem entre si, você pode usar a chave *networks* no arquivo *docker-compose.yml.*

4. Como configurar o container Nginx para atuar como um proxy reverso para redirecionar o tráfego para diferentes serviços dentro do Docker Compose?
   Para configurar o contêiner Nginx como um proxy reverso e redirecionar o tráfego para diferentes serviços dentro do Docker Compose, você precisa criar um arquivo de configuração do Nginx personalizado e montá-lo como um volume no contêiner Nginx.

5. Como especificar dependências entre os serviços no Docker Compose para garantir que o banco de dados PostgreSQL esteja totalmente inicializado antes do Python iniciar?
   Para especificar dependências entre os serviços no Docker Compose e garantir que o banco de dados PostgreSQL esteja totalmente inicializado antes do serviço Python iniciar, você pode usar a chave *depends_on* no

arquivo *docker-compose.yml.* No entanto, é importante observar que o *depends_on* apenas controla a ordem de inicialização dos contêineres, não garante a disponibilidade total do serviço dependente.

6. Como definir um volume compartilhado entre os containers Python e Redis para armazenar os dados da fila de mensagens implementada em Redis?
   Para definir um volume compartilhado entre os contêineres Python e Redis e armazenar os dados da fila de mensagens implementada em Redis, você pode usar a chave *volumes* no arquivo *docker-compose.yml.*

7. Como definir um volume compartilhado entre os containers Python e Redis para armazenar os dados da fila de mensagens implementada em Redis?
   Para configurar o Redis para aceitar conexões apenas de outros contêineres na rede interna do Docker Compose e não de fora, você pode ajustar a configuração do Redis para ouvir apenas no endereço IP da rede interna.

8. Como limitar os recursos de CPU e memória do container Nginx no Docker Compose?
   Para limitar os recursos de CPU e memória do contêiner Nginx no Docker Compose, você pode usar as opções *cpus* e *mem_limit* dentro do serviço correspondente no arquivo *docker-compose.yml.*

9. Como configurar o container Python para se conectar ao Redis usando a variável de ambiente correta especificada no Docker Compose?
   Para configurar o contêiner Python para se conectar ao Redis usando a variável de ambiente especificada no Docker Compose, você precisa definir a variável de ambiente no serviço Python e acessá-la no código Python.

10. Como escalar o container Python no Docker Compose para lidar com um maior volume de mensagens na fila implementada em Redis?
    Para escalar o contêiner Python no Docker Compose e lidar com um maior volume de mensagens na fila implementada em Redis, você pode usar a funcionalidade de escalabilidade do Docker Compose. A escalabilidade permite que você aumente o número de instâncias do serviço Python para distribuir a carga de trabalho.